1993

A Comparison of Queueing, Cluster and Distributed Computing Systems

Joseph A. Kaplan

Michael L. Nelson
Old Dominion University

Original Publication Citation

Kaplan, J. A., & Nelson, M. L. (1993). A comparison of queueing, cluster and distributed computing systems. *NASA Technical Memorandum: 109025*. Hampton, VA: NASA Langley Research Center

NASA Technical Memorandum 109025

A Comparison of Queueing, Cluster and Distributed Computing Systems

Joseph A. Kaplan and Michael L. Nelson
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia

October 1993

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virgina 23681-0001

A Comparison of Queueing, Cluster and Distributed Computing Systems

Joseph A. Kaplan
(j.a.kaplan@larc.nasa.gov)
Michael L. Nelson
(m.l.nelson@larc.nasa.gov)
NASA Langley Research Center
October, 1993

Abstract

Using workstations clusters for distributed computing has become popular with the proliferation of inexpensive, powerful workstations. Workstation clusters offer both a cost effective alternative to batch processing and an easy entry into parallel computing. However, a number of workstations on a network does not constitute a cluster. Cluster management software is necessary to harness the collective computing power. In this paper, we compare a variety of cluster management and queueing systems: Distributed Queueing Systems (DQS), Condor, LoadLeveler, Load Balancer, Load Sharing Facility (LSF - formerly Utopia), Distributed Job Manager (DJM), Computing in Distributed Networked Environments (CODINE) and NQS/Exec. The systems differ in their design philosophy and implementation. Based on published reports on the different systems and conversations with the system's developers and vendors, a comparison of the systems is made on the integral issues of clustered computing.

Introduction

Recently, there has been much interest in using inexpensive, powerful workstations to form workstation clusters (15,25,26). Workstation clusters offer many benefits over traditional central site computing. High performance workstation clusters can off-load jobs from saturated vector supercomputers, often providing comparable turn around time at a fraction of the cost. If workstations within clusters have a high speed interconnect, they may also serve as an inexpensive parallel computer. Several popular message passing systems such as PVM (19), Express (24), Linda (22) and P4 (23) enable users to tap the latent power of idle workstations by distributing their applications across the network.

However, a loose collection of workstations on the network (which may cross physical and political boundaries) does not constitute a cluster. Several terms have been used to define such activity: "cluster" (8), "farm" (25) and even "kluster" (21). Although cluster is an overloaded term, we prefer it to the alternatives. To avoid ambiguity, we define a "cluster" to be a collection of computers on a network that can function as a single computing resource through the use of additional system management software. The nature of the "additional system management software" and the expectations of its functionality are the focus of this report.

If workstation clustering is to be a real supplement to large scale computing, a cluster management system must be employed so that the total aggregate power of the cluster can be efficiently distributed across a wide user base. Workstation clusters are successfully in production at a number of sites serving both as a mid-range compute resource and as a parallel computer (15, 25). However, a clear "winner" has not emerged as a standard for cluster management. While some of the cluster management systems can concurrently co-exist on the same machines, at least for an initial evaluation period, this is not an acceptable long term solution. The needs of the cluster users must be evaluated, and then an appropriate cluster management system installed.

We compare eight popular distributed queueing and clustering systems. They are: Distributed Queueing System (DQS), Condor, LoadLeveler, Load Balancer, Load Sharing Facility (LSF - formerly Utopia), Distributed Job Manager (DJM), Computing in Distributed Networked Environments (CODINE) and NQS/Exec. These systems, while not an exhaustive list, represent a broad spectrum of functionality, and not all features map from one system to another. A set of criteria was constructed to facilitate the comparison of these systems. The list of criteria is intended to be general, but it cannot be expected to cover the concerns of every user community.

Evaluation Criteria

1.0 Heterogeneous Computing Environment

There are two types of cluster environments, homogeneous and heterogeneous. A homogeneous computing environment consists of a number of computers of the same architecture running the same operating system. A heterogeneous computing environment consists of a number of computers with dissimilar architectures and different operating systems. Many locations have a large number of different computers for varying resource and performance considerations (i.e. graphics, computation, input/output operation, etc.).

1.1 Which platforms are supported?

If the clustering system does not support popular machines at a given site, it will be of limited use. By examining the platforms supported and the existing machine base, a suitable match can be made. Given enough customer response, vendors may be willing to include support for additional platforms. A listing of release and revision numbers that are supported for each operating system is not included since this information is so dynamic.

1.2 How does the system know upon which architecture to execute the job?

A heterogeneous environment creates additional complexity for program execution. This is a description of how the clustering system determines the correct target architecture and locates the necessary files and resources for efficient execution.

1.3 What additional hardware and/or software is required for the system to work?

In order to function as advertised, many packages require additional software or hardware to be

installed. Examples include Network File System (NFS), Andrew File System (AFS), or the Name Information Server (NIS). Any significant modifications needed for the existing platform constitute an additional burden upon the system manager. These items also represent unforeseen costs. Even if a cluster management software is free, it may require commercial software to run properly (2,8,9,11,12,13).

2.0 Application Support

In today's multi-disciplinary research environment, a variety of approaches for performing computational tasks are required. They include parallel distributed applications, interactive sessions, and traditional batch computation. A clustering management system must support both the present and developing methods of computation (14,15).

2.1 Is support provided for batch jobs?

A popular use of clusters is for off-loading batch jobs from saturated supercomputers. Clusters can often provide better turn around time than supercomputers for small (in terms of memory and CPU requirements) batch jobs.

2.2 Is support provided for parallel jobs?

There is interest in moving to massively parallel processing machines via heterogeneous processing because of the heterogeneous environment's application to a larger set of problems (14,15,16). A cluster can serve as a parallel machine because workstations are inexpensive and easier to upgrade as separate pieces may be purchased to replace older models. A number of packages, such as Parallel Virtual Machine (PVM) from Oak Ridge National Laboratories and Express from Parasoft Inc. (19,20), add parallel support for computers distributed across a network. References 16 and 27 provide a good overview of the various tools available for parallel application development.

2.3 Is support provided for interactive jobs?

One of the tasks a cluster should provide is the ability for users to execute interactive jobs on the cluster. The input, output, and error messages should all be optionally returned to the user's interactive machine.

2.4 Is support provided for message passing between jobs?

Message passing is the ability to pass data between processes in a standardized method. This inter-process communication allows several processes to work on a single problem in parallel. A large distributed application can be split across the many different platforms that exist in a heterogeneous environment. Some cluster management packages do not provide any explicit message passing support, choosing instead to rely on packages such as PVM and Linda to provide that feature (19,20).

2.5 Can the system handle redirective jobs (i.e. stdin, stdout, stderr)?

The ability to redirect I/O files is very similar to the Unix redirection commands available from the command line (i.e. >,>>,<,<<,etc.). This feature allows users to run existing interactive jobs in a batch environment.

2.6 Can the files be accessed from the submitting machine?

The ability of the user to access the output files from the submitting machine is a significant step in providing an easy to use system. The user should not be concerned with the details of physical file location. Files should be visible from the relevant machines or should migrate to necessary machines transparent to the user.

3.0 Scheduling and Resource Allocation

As jobs are added to the system, it is important to know how, where and when the jobs will be scheduled. If the scheduling is done improperly, jobs will not be processed as efficiently as possible. The method chosen will greatly affect the performance and efficiency of the system.

3.1 What is the dispatching of jobs based upon?

A wide combination of system attributes must be examined before a job is dispatched in order to generate an optimal balance. These attributes can include system load, computational power of the destination machine, physical proximity, degree of owner control and organizational location.

3.1.1 Users

The cluster system software should recognize when a cluster machine is in use and schedule jobs for that machine so as to not greatly impact the current users, both console and remote. This can include suspending or migrating jobs to other available machines.

3.1.2 Disk Space

Batch jobs often write large data files while executing. If the system supports a checkpointing mechanism, the executing job can consume a tremendous amount of disk space (3). A job should not be dispatched if there is insufficient disk space on the destination machine.

3.1.3 System Load

The goal of cluster management software is to balance the load on all of the machines in order to harvest the idle CPU cycles. Executing a job on a lightly loaded CPU will enable it to complete execution as quickly as possible. If a machine is overloaded with jobs, it will not produce its maximum computational power. A careful balance must be maintained in order to achieve maximum cluster performance.

3.1.4 Swap Space

Swap space is the portion of the disk space that implements virtual memory. Jobs moved from supercomputers to clusters may have large memory requirements that represent a significant portion of available memory of a cluster machine. Thus, the cluster management software should not dispatch a job to a machine with insufficient swap space.

3.2 Does the system provide checkpointing facilities for jobs?

Checkpoint is a common method used by cluster management software to save the current state of the job (3). In the event of a system crash, the only lost computation will be from the point at which the last checkpoint file was made. Because checkpointing in a heterogeneous environment is more difficult than on a single architecture, current cluster management software that provides checkpointing does so with the following limitations:

- Only single process jobs are supported (i.e. no fork(), exec(), or similar calls are allowed).
- No signals or signal handlers are supported (i.e. signal(), sigvec(), and kill() are not supported).
- No interprocess communication (i.e. sockets, send(), recv(), or similar calls are not implemented).
- All file operations must either be read only or write only. These limitations will make checkpointing unsuitable for certain applications, including parallel or distributed jobs that must communication with other processes.

3.3 Is process migration between machines in the cluster supported?

Process migration is the ability to move a process from one machine to another machine without restarting the program, thereby balancing the load over the cluster. Process migration would ideally be used if the load on a machine becomes too high, or someone logs on to the console, thus allowing processes to migrate to another machine and finish faster.

3.4 What is the impact on the machine owner?

An economical approach to implementing or supplementing a cluster is to use existing machines as opposed to purchasing new ones. One method of insuring the growth of the cluster is to guarantee that owners of clustered workstations will not experience unacceptable loads on their system while they are working interactively.

3.5 Are changes made in scheduling to account for fault tolerance?

In order to detect whether a host is running, the master process communicates with all the cluster members by sending messages over the network. When the master process has not heard from a member after a set period of time, it assumes that the member has died. The system should no longer schedule jobs to that machine. A balance must be obtained to ensure that crashes are detected early, but without generating too much network traffic determining which machines are still alive.

3.6 Does the system have the ability to suspend and/or resume jobs when resources change or become unavailable?

Most systems can suspend all currently running jobs on a machine if the owner logs onto the console. The suspended jobs can then be resumed when the machine becomes idle or migrated to another machine while the owner continues to use the machine. Jobs should also be able to suspend in the dispatch queues if they can only execute on a particular machine that is currently unavailable.

4.0 Configurability

Since each computing environment is unique, cluster management systems must be flexible. The ability to modify system parameters and attributes by the system administrator and/or user makes the system more adaptable to its environment.

4.1 Is it possible to place restrictions on the users of the system?

The clustering system should allow for users to have dissimilar access to system resources.This includes limiting: CPU time, amount of memory, swap space, disk space, ability to execute certain jobs, ability to submit jobs to certain machines, queues, architectures, and time of day access.

4.2 Does the system enforce job run time limits?

A run time limit details the amount of CPU time a job is allowed for execution. Providing a CPU limit insures that smaller jobs can complete without a prolonged delay incurred by waiting behind a job that runs for an excessive period of time. While run time limits do offer many benefits to the user, the limits must be properly configured for the users' needs and environment.

4.3 Are priorities implemented for scheduling and resource allocation?

In addition to enabling more efficient job execution, priorities allow a clustering scheduler to respect time critical applications and political boundaries.

4.4 Can an exclusive CPU be made available if a job requires it?

An exclusive CPU gives the ability to run only one program on a single CPU. This can be done by waiting until all running jobs complete or having the jobs currently running suspend. Jobs that have special requirements might need an exclusive CPU, such as benchmarking the machine or a program, a real-time environment, or to provide minimum turn around time for an application.

4.5 Can the user or system manager defer execution of a job?

As users leave the office at the end of the work day and on the weekends, more resources become available for computational use. Delaying execution of computationally intensive jobs can pro-

vide faster turn around for smaller jobs. Users that submit large numbers of jobs can delay execution of their work until more resources become available.

4.6 Does the system have a Graphical User Interface (GUI)?

A well designed Graphical User Interface (GUI) can aid in guiding users through the system. If users are frustrated, or do not understand the system due to a non-intuitive or over complicated interface, the system will be underutilized.

4.7 Is the system easy to learn?

If the users are unable to easily and quickly learn the commands necessary to use the system, the potential of the cluster will not be realized. An interface similar to NQS will facilitate the transition of jobs from supercomputers to clusters.

4.8 Can the user specify computing resources?

In order for each job to be executed on the machine to achieve optimal performance, user customizable allocation is needed. These desired resources can be specified in a GUI, a job submission file or on a command line interface. If users specify too large a resource, their jobs will be delayed until those resources become available. These resources include the individual machine, a particular architecture, the amount of memory, the amount of swap space, the type of operating system, and the amount of disk space.

4.9 Can users query the status of their jobs?

Once users submit a job, they no longer directly control it. In order to keep them updated about the status of their job, they should have some method of monitoring the job's status. This can be either through e-mail messages or an interactive program.

4.10 Does the system return actual resource usage upon job completion?

If users can see the actual resources that a job uses, they will be better able to estimate the requirements for the next time the job executes. These statistics will aid the user in providing for optimal job performance.

5.0 Dynamics of the system

In a dynamic heterogeneous computing environment, the make up of the cluster can change. Machines will be taken down for maintenance. Other machines will be removed from the cluster for dedicated use. The system must be able to be changed dynamically by either the system manager or the system itself. It must constantly adapt to the resources that are available.

5.1 Can the policies and queues of the system be reconfigured on the fly?

A cluster size can vary from a few machines to thousands of machines (13). The cluster administrator should have the ability to reconfigure the cluster without restarting the processes on every machine. However, the cluster should not significantly increase network traffic attempting to determine if the configuration has changed.

5.2 Can queues or machines be time of day sensitive?

Workstations often become idle during non-prime hours. If the cluster can gain use of these machines automatically at the end of the work day, during the weekend, or whenever they are idle, these extra resources can be utilized.

5.3 Can the machine be donated and withdrawn at will by the principal owner without the help of the system manager?

The configuration of a cluster can change dynamically. Users might want to withdraw their machines for some particular job they are planning to run. Machines may also be added for special one time jobs. The cluster should not have to be constantly reconfigured. The principal owner of the machine should have the ability to withdraw their machine without requiring root access or help from the system manager.

5.4 Does the system possess any single point failure problems?

A single point failure problem can make a cluster unsuitable for some applications. A common single point failure problem for cluster management software is in the master scheduler.

5.5 Is the system fault tolerance?

The software should be able to withstand rapid changes in the availability of machines and/or communications. If the scheduling software crashes, the rest of the cluster management package should be robust enough to continue to function. If the system crashes, the programs will not complete. All lost computation represents lost productivity. The system should be able to guarantee to the user that submitted jobs will eventually complete.

5.6 How does the system address security?

Participating in a cluster should not comprise the security of the user's workstation. Cluster management software should not introduce any security holes. Some systems prefer to simply use the security features that are provided with the operating system while other implement stronger means of defeating intentional or unintentional breaches of security.

Evaluations

The next eight following sections contain reviews of each package with respect to the criteria described above. Each section is designed so that it stand alone. Table 1 provides a quick reference chart so that the criteria highlights can be easily compared against each system.

Figure 1 provides a graphical depiction of the relationships between the different systems. The Network Queueing System (NQS) is considered to be the progenitor of all the systems. If an arrow continues outward, the system is still in production and development. If a system feeds linearly into another, it indicates a name change. If a system branches from another's line, it indicates that the new system has evolved from the former system.

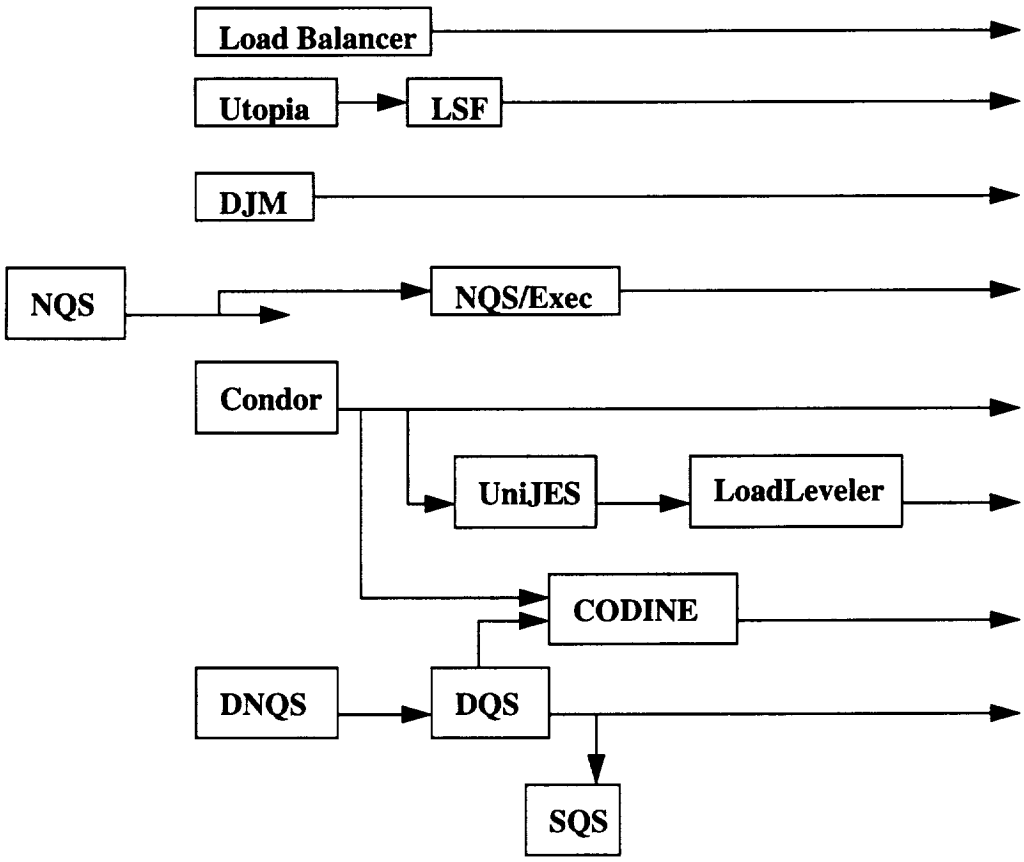

Figure 1: A Pictorial Clustering System Ancestry

Table 1: Quick Reference Chart

Attributes	Condor	DQS	DJM	LoadLeveler	Load Balancer	LSF	CODINE	NQS/Exec
Heterogenous support	X	X		X	X	X	X	X
Interactive support		X	X		X	X	X	X
Batch support	X	X	X	X	X	X	X	X
Parallel support		X	X			X	X	
Checkpointing	X			X			X	
Process Migration	X			X			X	
Job run-time limit		X	X	X	X	X	X	X
No single point failure	X		X	X		X		X
Defer Execution			X	X	X	X	X	X
GUI		X		X			X	X
NFS/AFS not Required	X							
Withdraw Machine at will		X		X			X	
Return resources used	X	X	X	X	X	X	X	X
Logins not needed	X							
Time of day sensitive			X	X	X	X	X	X
I/O redirection	X	X	X	X	X	X	X	X
PublicDomain	X	X	X					

Computing in Distributed Networked Environments(CODINE)

Concept behind Implementation

CODINE (COmputing in DIstributed Networked Environments), a merger between the Condor and DQS packages, is targeted for optimal utilization of the compute resources in heterogeneous networked environments, in particular, large heterogeneous workstation clusters with integrated compute servers like vector and parallel supercomputers. CODINE offers a uniform, convenient, easy to use and easy to administer batch queueing framework for a large variety of architectures. As such, CODINE provides dynamic and static load balancing, checkpointing, and supports batch, interactive and parallel jobs. A graphical user interface and various accounting and utilization statistics enhance the usability of CODINE.

1.0 Is support provided for a heterogeneous computing environment?

Yes.

1.1 Which platforms are supported?

ConvexOS, Cray UNICOS, DEC OSF/1 - Ultrix, HP-UX, IBM AIX, SGI IRIX, and SunOS - Solaris.

1.2 How does the system know upon which architecture to execute the job?

Specified by the user either on the command line or in the job submission file.

1.3 What additional hardware and/or software is required for the system to work?

NFS and NIS are recommended.

2.0 Application Support

2.1 Is support provided for batch jobs?

Yes.

2.2 Is support provided for parallel jobs?

Yes. This support is provided through PVM, Express, P4, and Linda.

2.3 Is support provided for interactive jobs?

Yes. Support is provided through an interactive xterm.

2.4 Is support provided for message passing between jobs?

Yes. Through PVM, Express, P4, and Linda.

2.5 Can the system handle redirective jobs?

No.

2.6 Can the files be accessed from the submitting machine?

Yes. This can be done through NFS.

3.0 Scheduling and Resource Allocation

Jobs may be scheduled according to two algorithms. The first algorithm schedules the job on the least loaded host at the time suitable queues are available to start the job. The second algorithm dispatches the job to a queue according to a given order defined by the cluster manager.

3.1 What is the dispatching of jobs based upon?

User specification of desired resources.

3.1.1 Users

Utilities can be run that check for interactive users.

3.1.2 Disk Space

No.

3.1.3 System Load

Yes, depending upon the scheduling algorithm chosen. See section 3.0.

3.1.4 Swap Space

No.

3.2 Does the system provide checkpointing facilities for jobs?

Yes; however, programs with fork(), exec(), and IPC-calls(i.e. socket(), etc.) will not checkpoint.

3.3 Is process migration between machines in the cluster supported?

Yes.

3.4 What is the impact on the machine owner?

The user may suspend all activity on their machine by using the qmod command or by running qidle, which checks for X Window System activity.

3.5 Are changes made in scheduling to account for fault tolerance?

Yes. CODINE will no longer send jobs to a machine that is not in communication with the master scheduler.

3.6 Does the system have the ability to suspend and/or resume jobs when resources change or become unavailable?

Yes. This is done through checkpointing and migration.

4.0 Configurability

4.1 Is it possible to place restrictions on the users of the system?

Yes. Global and queue access to the system may be allowed or denied. Certain users may be provided with queue ownership that allows them to suspend or enable certain queues.

4.2 Does the system enforce job run time limits?

Yes.

4.3 Are priorities implemented for scheduling and resource allocation?
Yes

4.4 Can an exclusive CPU be made available if a job requires it?
No.

4.5 Can the user or system manager defer execution of a job?
Yes.

4.6 Does the system have a Graphical User Interface (GUI)?
Yes. CODINE provides a motif based GUI.

4.7 Is the system easy to learn?
Yes. The system is very similar to NQS.

4.8 Can the user specify computing resources?
Yes.

4.9 Can the user query the status of their job?
Yes. This is done by using a utility supplied with CODINE or by using the GUI.

4.10 Does the system return actual resource usage upon job completion?
Yes.

5.0 Dynamics of the system

5.1 Can the policies and queues of the system be reconfigured on the fly?
Yes.

5.2 Can the queues or machines be time of day sensitive?
Yes.

5.3 Can the machine be donated and withdrawn at will by the principal owner without the help of the system manager?
Yes.

5.4 Does the system possess any single point failure problems?
Yes. The master scheduler must be reachable by every machine in the cluster. If this machine fails, the cluster will not function.

5.5 What conditions has the system made for fault tolerance?
CODINE offers checkpointing and migration for fault tolerance. Any jobs running on a crashed machine will be restarted on another host. Dependence upon a single master scheduler will be changed in a future release.

5.6 How does the system address security?
Global and queue specific user access lists and modes are maintained and Unix security mechanisms are supported. Future releases will integrate Kerberos.

Additional Limitations
• The software has not been widely distributed.
• Applications with process creation (fork(), exec()), signals (signal(), sigvec(), and kill()) and Interprocess Communication (IPC) will not checkpoint.

Additional Benefits
• CODINE attempts to merge the best features of Condor and DQS.

Sites running this software:
Volkswagon
Max-Planck-Institutes
University of Munich
University of Regensburg

Software From:
Dr. Wolfgang Gentzsch
GENIAS Software GmbH
Erzgebirgstr. 2, W-8402 Neutraubling b.
Regensburg, Germany
Tel+49 9401 92000
Fax+49 9401 920092
e-mail gent@genias.de
See Reference: 18, 28

Condor

Concept behind Implementation

Condor is built on the principal of distributing batch jobs around a loosely coupled cluster of computers. Computational tasks are rarely distributed evenly across desktop workstations, with most workstations idling while some are fully loaded. Condor attempts to use idle cpu cycles that exist on some machines to help in providing more computational power to users. There are three main guiding principles that Condor attempts to follow while doing this. The first is that the workstation owner should always have their workstation at their disposal when they want it. Users will be less willing to donate their idle cpu cycles if they cannot have instant access to their machine. The second principal is that remote execution should be easy. If the user has to go through too much trouble, they will not use the system. The last principle is that is should be portable, and easy to adapt to emerging workstation platforms.

1.0 Is support provided for a heterogeneous computing environment?

Yes.

1.1 Which platforms are supported?

DEC OSF/1 - Ultrix, HP-UX, IBM AIX, Sequent Dynix, SGI IRIX, and SunOS - Solaris.

1.2 How does the system know upon which architecture to execute the job?

The user can specify which architecture in a job description file that is submitted with the job. If no description is given, then the architecture defaults to the type of the submitting machine.

1.3 What additional hardware and/or software is required for the system to work?

NFS mounting capabilities are strongly recommended, but not necessary.

2.0 Application Support

2.1 Is support provided for batch jobs?

Yes.

2.2 Is support provided for parallel jobs?

No. However, PVM support is currently being added.

2.3 Is support provided for interactive jobs?

No.

2.4 Is support provided for message passing between jobs?

No message passing support is implemented at this time. Support for PVM is currently being added by the authors.

2.5 Can the system handle redirective jobs?

Yes. This is specified by the user in the job submission file.

2.6 Can the files be accessed from the submitting machine?
Yes. All input/output is done through the submitting machine using either NFS or remote system calls.

3.0 Scheduling and Resource Allocation
The condor software monitors the activity on all the participating workstations in the local network. Those machines which are determined to be idle are placed into a resource pool or "processor bank." Machines are then allocated from the bank for the execution of jobs by the scheduler. The bank is a dynamic entity; workstations enter the bank when they become idle, and leave again when they get busy. Priority is based on the up-down algorithm. This algorithm periodically increases the priority of those users who have been waiting for resources, and reduces the priority of those users who have received resources in the recent past. This is done by the centralized machine manager.

3.1 What is the dispatching of jobs based upon?
Condor jobs are not submitted to a machine that has an active keyboard or mouse or a machine that has a load greater than 0.3. This default load average can be configured by the system administrator or by the owner of each machine.

3.1.1 Users
Yes. If the console is active, no condor jobs will be dispatched. All condor activity currently on the machine will be suspended.

3.1.2 Disk Space
Yes.

3.1.3 System Load
Yes. The load must be below 0.3 before condor will submit any jobs to the machine

3.1.4 Swap Space
Yes.

3.2 Does the system provide checkpointing facilities for jobs?
Yes. However, programs with fork(), exec(), and IPC-calls(i.e. socket(), etc.) will not checkpoint.

3.3 Is process migration between machines in the cluster supported?
Yes. Checkpoint files are migrated after a predefined time period if user activity continues on the console.

3.4 What is the impact on the machine owner?
Very little. Condor attempts to minimize any adverse effects for the machine owner. It does this by migrating away tasks as soon as the machine owner begins work at the console.

3.5 Are changes made in scheduling to account for fault tolerance?
Yes. Condor will only submit jobs to machines that are running.

3.6 Does the system have the ability to suspend and/or resume jobs when resources change or become unavailable?
Yes. Scheduling changes are based upon user activity.

4.0 Configurability

4.1 Is it possible to place restrictions on the users of the system?
No.

4.2 Does the system enforce job run time limits?
No.

4.3 Are priorities implemented for scheduling and resource allocation?
Yes. The user can specify a priority for their job in the job submission file.

4.4 Can an exclusive CPU be made available if a job requires it?
No.

4.5 Can the user or system manager defer execution of a job?
No.

4.6 Does the system have a Graphical User Interface (GUI)?
No.

4.7 Is the system easy to learn?
Yes. The user must relink the programs with a special condor library, but the jobs submission files are very easy to produce.

4.8 Can the user specify computing resources?
Yes. The user may specify the machine, the type of architecture, the amount of memory, the amount of swap space, and the operating system. These items are specified in the job submission file.

4.9 Can the user query the status of their job?
Yes. This is done through a utility supplied with condor.

4.10 Does the system return actual resource usage upon job completion?
Yes. This information is sent by e-mail. The person who will receive the message is specified in the job submission file.

5.0 Dynamics of the System

5.1 Can the policies and queues of the system be reconfigured on the fly?
No. The condor resource pool changes as user activity changes. When machines fall idle, they are

added to the condor pool if they have the condor software installed on them.

5.2 Can the queues or machines be time of day sensitive?
No. Condor's scheduling changes based upon user activity.

5.3 Can the machine be donated and withdrawn at will by the principal owner without the help of the system manager?
Condor will automatically withdraw the machine when it becomes active by the owner. Withdrawing the machine permanently would require removing the scheduling daemon. This will require someone with root access.

5.4 Does the system possess any single point failure problems?
No. If a part of the system goes down, the rest of the system will continue to function. If the master scheduler goes down, it will need to be restarted before more jobs will be dispatched.

5.5 What conditions has the system made for fault tolerance?
No jobs loss, other than the work done between checkpoints, will occur if anything crashes. If the daemon dies, then everything should be able to be restarted without difficulty.

5.6 How does the system address security?
Condor attempts to maintain the security of unix, but it does have some problems. The remote system calls are a weak point in the system that can be manipulated by a malicious user.

Additional Limitations
• Applications with process creation (fork(), exec()), signals (signal(), sigvec(), and kill()) and Interprocess Communication (IPC) will not checkpoint. These calls should be supported in the next version of Condor.

• All file operations must be idempotent - read-only and write-only file accesses work correctly, but programs which both read and write the same file may not work correctly when using checkpointing.

• Each condor job has an associated checkpoint file which is approximately the size of the address space of the process. Disk space must be available to store the checkpoint file both of the submitting and executing machines.

Additional Benefits
• The software is available in the public domain.

• There is a mailing list set up to offer help for condor users and system managers.

Sites running this software:
NASA Lewis Research Center
Brookhaven National Laboratory
University of Wisconsin

Software Available from:
anonymous ftp: ftp.cs.wisc.edu:/condor
See References: 1,2,3,4,5

DJM (Distributed Job Manager)

Concept behind Implementation

Distributed Job Manager, DJM, is a drop in replacement for Network queueing System (NQS). It was written by the Minnesota Supercomputer Center and is made freely available under the GNU copyright, but the GNU project has no ownership of the software. DJM only runs on Thinking Machines Corporation's Connection Machine-2, CM-2, and Connection Machine-5, CM-5, massively parallel machines. DJM's main design requirements are (1) Be upwardly compatible with NQS, (2) Provide full support for interactive use, (3) Balance job load across the machine, and (4) Provide flexible controls and fault tolerance.

1.0 Is support provided for a heterogeneous computing environment?

No. Support is only provided for Thinking Machines Corporation's architectures.

1.1 Which platforms are supported?

Thinking Machines Corporation's CM-2 and CM-5.

1.2 How does the system know upon which architecture to execute the job?

The job will run on the machine that it is submitted to.

1.3 What additional hardware and/or software is required for the system to work?

The system requires gcc version 2.1 or later. One or more SUN or SGI computers are needed to front-end the Connection Machine.

2.0 Application Support

2.1 Is support provided for batch jobs?

Yes.

2.2 Is support provided for parallel jobs?

Yes. Since DJM only supports CM-2 and CM-5 machines, it supports parallel jobs.

2.3 Is support provided for interactive jobs?

Yes.

2.4 Is support provided for message passing between jobs?

Yes.

2.5 Can the system handle redirective jobs?

Yes. Specifications for this is given in the job submission file.

2.6 Can the files be accessed from the submitting machine?

Yes. This can be accomplished either through NFS or AFS.

3.0 Scheduling and Resource Allocation

3.1 What is the dispatching of jobs based upon?
DJM will dispatch a job to the least loaded partition that satisfies all of the job's resource requirements.

3.1.1 Users
No.

3.1.2 Disk Space
No.

3.1.3 System Load
Yes. DJM dynamically monitors the load of the machine and the front-ends.

3.1.4 Swap Space
No.

3.2 Does the system provide checkpointing facilities for jobs?
No. While this feature is not presently supported, it is hoped that it will be added in the future.

3.3 Is process migration between machines in the cluster supported?
No. While this feature is not presently supported, it is hoped that it will be added in the future.

3.4 What is the impact on the machine owner?
The CM-2 and CM-5 computers are not considered desktop machines and therefore aren't considered to have a single owner.

3.5 Are changes made in scheduling to account for fault tolerance?
Yes. Queued jobs will be rerouted to other partitions until the crashed partition is usable. The jobs that were running on the partition that failed will be rerun on other partitions.

3.6 Does the system have the ability to suspend and/or resume jobs when resources change or become unavailable?
No. While this feature is not presently supported, it is hoped that it will be added in the future.

4.0 Configurability

4.1 Is it possible to place restrictions on the users of the system?
Yes. This is done by providing both user and group limits. Per-project limits are also provided. Projects are charging categories, used mainly for billing or accounting purposes.

4.2 Does the system enforce job run time limits?
Yes. DJM provides user, group, and queue job run time limits.

4.3 Are priorities implemented for scheduling and resource allocation?
Yes. Priorities can be established for individual users, for groups of users, and for each queue.

4.4 Can an exclusive CPU be made available if a job requires it?
Yes. Partitions can be established in such a way that only one job can use it. Partitions are groups of processors. Since each machine can have from sixteen to thousands of processors, the number of exclusive partitions can be large.

4.5 Can the user or system manager defer execution of a job
Yes. The user has an option to defer execution until a specified time or day. The operator may also place a hold on a job and then release it later.

4.6 Does the system have a Graphical User Interface (GUI)?
No.

4.7 Is the system easy to learn?
Yes. DJM is similar to NQS.

4.8 Can the user specify computing resources?
Yes. The user can specify the machine, the amount of memory, and the amount of disk space. This can be done as a command line argument, in a job submission file, or the user can submit the job to a partition that has the required amount of resources.

4.9 Can users query the status of their job?
Yes. This is done interactively by the user through a utility supplied with DJM.

4.10 Does the system return actual resource usage upon job completion?
Yes.

5.0 Dynamics of the System

5.1 Can the policies and queues of the system be reconfigured on the fly?
Yes. All configuration information is stored in an ascii format that can be modified with any text editor. The system checks this file at a defined interval to determine is the configuration has changed. If a change is detected, the system reconfigures itself according to the specifications set forth in the ascii file.

5.2 Can the queues or machines be time of day sensitive?
Yes. Partitions can become dedicated or may become shared at certain time spans.

5.3 Can the machine be donated and withdrawn at will by the principal owner without the help of the system manager?
No. As stated in section 3.4, there is usually no principal owner of a CM-2 or CM-5.

5.4 Does the system possess any single point failure problems?
No. A portion of the CM machine or a front-end can go down without adversely affecting other

components of the system.

5.5 Is the system fault tolerance?
The system is very robust in the manner that is handles crashed partitions. Jobs are routed around crashed partitions at the same time that the jobs from the crashed partition are being rerun.

5.6 How does the system address security?
DJM uses an authentication system based on passing of encrypted credentials. Security attacks against the job system result in an alarm message being sent to the system administrator.

Additional Limitations
• DJM is only relevant if a Connection Machine is to be used. It is included here for comparison purposes.

Additional Benefits
• DJM is very similar to NQS and the job scripts do not need to be rewritten.

• DJM is free and there is a mailing list to obtain help or answer questions should they arise.

• DJM is designed to support a massively parallel system. The source code is freely available and users are welcome to try to port it for their own use.

Sites running this software:
NASA Ames Research Center

Software from:
anonymous ftp: ec.msc.edu:pub/LIGHTNING
See References: 6

DQS (Distributed Queueing System)

Concept behind Implementation

DQS was written at the Supercomputer Computations Research Institute (SCRI) at Florida State University with collaborators at The Center For High Performance Computing (CHPC) at The University of Texas and The Pittsburgh Supercomputing Center (PSC). DQS is popular because it is the first non-commercial clustering system that is conceptually similar to NQS. DQS provides the user with different queues based upon architecture and group. All jobs are submitted to individual queues to await execution. DQS was originally Distributed Network Queueing System (DNQS). McGill University performed a rewrite of DNQS and renamed it Dynamical Network Queueing System[1]. Because the source code is available, many variants of DQS exist. This review is relevant only to DQS as distributed from SCRI, CHPC and PSC. Additionally, an unpublished report from SCRI suggested a future product, the Scalable Queueing System (SQS), but this project has not begun.

1.0 Is support provided for a heterogeneous computing environment?

Yes.

1.1 Which platforms are supported?

ConvexOS, Cray UNICOS, DEC OSF/1, HP-UX, IBM AIX, SGI IRIX, and SunOS - Solaris.

1.2 How does the system know upon which architecture to execute the job?

Either a queue for a particular system or a group queue for a class of systems must be specified by the user in the job submission file.

1.3 What additional hardware and/or software is required for the system to work?

Either AFS or NFS is required for the system to work.

2.0 Application Support

2.1 Is support provided for batch jobs?

Yes.

2.2 Is support provided for parallel jobs?

Yes. Parallel jobs may be run under PVM.

2.3 Is support provided for interactive jobs?

Yes. Interactive queues offer xterm sessions on their respective machines.

2.4 Is support provided for message passing between jobs?

Yes. Support is provided for PVM 3.1 and 2.4.

---

1. Dynamical Network Queueing System is still available anonymous FTP from ftp.physics.mcgill.ca in directory: /pub/Dnqs. We consider DQS to have superceded DNQS.

2.5 Can the system handle redirective jobs?
Yes. This may be specified by the user on the command line or in a job submission file.

2.6 Can the files be accessed from the submitting machine?
Yes. The system must be set up using either NFS or AFS.

3.0 Scheduling and Resource Allocation
There are two methods of scheduling possible. The first is to schedule jobs according to the queue sequence number (i.e. the first queue in the list receives the job for execution).The second method is to schedule by weighted load average within a group so that the least busy node is selected to run the job. The actual method used is selected at compilation time.

3.1 What is the dispatching of jobs based upon?
User specification of resources.

3.1.1 Users
No. However, if an interactive user logs onto a machine, all jobs currently running can be suspended until the machine becomes idle.

3.1.2 Disk Space
No.

3.1.3 System Load
Yes. DQS can be configured to send the job to the machine in the requested queue with the lightest system load

3.1.4 Swap Space
No.

3.2 Does the system provide checkpointing facilities for jobs?
No.

3.3 Is process migration between machines in the cluster supported?
No.

3.4 What is the impact on the machine owner?
Very little. DQS can suspend the queue if the console becomes active (i.e. keyboard or mouse activity).

3.5 Are changes made in scheduling to account for fault tolerance?
Yes. If the master scheduler is unable to contact a machine in a set period of time, it will no longer attempt to schedule any jobs to that node.

3.6 Does the system have the ability to suspend and/or resume jobs when resources change or become unavailable?
Yes. If a down machine comes back on-line, it will contact the master scheduler and its jobs will be rescheduled. If a user logs on to one of the workstations, the queue can be suspended. When the machine falls idle, its queue will be restarted, including the jobs in the queue if so configured.

4.0 Configurability

4.1 Is it possible to place restrictions on the users of the system?
Yes. Each queue can configured at different levels. Users can then be restricted or admitted to each queue.

4.2 Does the system enforce job run time limits?
Yes. Each queue may have a per-process run time limit or a per-job run time limit.

4.3 Are priorities implemented for scheduling and resource allocation?
Yes. Queues can be set up on specific machines with special resources that cannot be accessed without accessing the queue. Queues may also be set up with different priorities.

4.4 Can an exclusive CPU be made available if a job requires it?
Yes. Queues can be suspended when other queues receive jobs.

4.5 Can the user or system manager defer execution of a job?
No.

4.6 Does the system have a Graphical User Interface (GUI)?
Yes.

4.7 Is the system easy to learn?
Yes. DQS is conceptually similar to NQS.

4.8 Can the user specify computing resources?
Yes. The user may request resources either through a job submission file, on the command line, or through the GUI. The user may request a specific machine, a specific architecture, the amount of memory, or a specific operating system.

4.9 Can the user query the status of their job?
Yes. This can be done interactively with the GUI or with a utility, both of which are supplied with DQS.

4.10 Does the system return actual resource usage upon job completion?
Yes, but only for single node jobs. Metrics for parallel jobs are not reported with the exception of the master host.

5.0 Dynamics of the System

5.1 Can the policies and queues of the system be reconfigured on the fly?
Yes. Queues can be added, deleted, and reconfigured while the system is in operation. Hosts may be added and deleted, the global cluster configuration may be modified, and users may be added and deleted from the access control database without restarted the system.

5.2 Can the queues or machines be time of day sensitive?
No; however, this feature will be added in future releases.

5.3 Can the machine be donated and withdrawn at will by the principal owner without the help of the system manager?
Yes, but DQS must be configured by the system manager to allow this to happen.

5.4 Does the system possess any single point failure problems?
Yes. The machine which runs the master scheduler represents a single point failure problem. A future release of DQS will allow multiple qmasters.

5.5 What conditions has the system made for fault tolerance?
Other than the problem addressed in section 5.4, the system is very stable.

5.6 How does the system address security?
DQS relies on the security provided by the Unix operating system.

Additional Limitations
• User must have files on the machine that will execute the jobs. This can be done through NFS. If the user's job cannot access the necessary files or does not have a login on the executing machine, the job fails without sending any type of error message to the user.

Additional Benefits

• DQS is popular and well supported by its development team.

Sites running this software:
Boeing
SCRI (Florida State University)
PSC
CHPC
Hewlett Packard
Apple Computer

Software From:
anonymous ftp: ftp.scri.fsu.edu:/pub/dqs
See References: 7,8

Load Balancer

Concept behind Implementation

Load Balancer automatically queues and distributes jobs across a heterogeneous network of UNIX computers. It improves performance by putting idle computers to work, and reducing the load on busy ones. Load Balancer tries to eliminate resource contention (CPU/memory/swap) by queueing excess jobs. Load Balancer is built upon the premise of compute servers.

1.0 Is support provided for a heterogeneous computing environment?

Yes.

1.1 Which platforms are supported?

DEC OSF/1 - Ultrix, HP-UX, IBM AIX, SGI IRIX, and SunOS.

1.2 How does the system know upon which architecture to execute the job?

The user may specify the architecture desired on the command line when the job is submitted. If the user does not specify the architecture, then the system will default to either the user, application, or system default specified in the configuration file.

1.3 What additional hardware and/or software is required for the system to work?

NFS and NIS are recommended.

2.0 Application Support

2.1 Is support provided for batch jobs?

Yes.

2.2 Is support provided for parallel jobs?

No. Support for parallel jobs is planned for the future.

2.3 Is support provided for interactive jobs?

Yes.

2.4 Is support provided for message passing between jobs?

Yes. Load Balancer does not require modification to the application; it is not concerned with the application's functionality.

2.5 Can the system handle redirective jobs?

Yes. This is done through the command line interface.

2.6 Can the files be accessed from the submitting machine?

Yes. The files must be present on both the submitting and executing machine. This can be accomplished through NFS or AFS.

3.0 Scheduling and Resource Allocation

Jobs are submitted to a priority based queue. As resources become available, the jobs are sent out the to the individual machines. Load Balancer attempts to keep the load of each machine at an appropriate level. Faster machines will receive more work that slower machine, and a proportional work load should result.

3.1 What is the dispatching of jobs based upon?

The goal of the scheduler is to maintain appropriate loads on individuals machines, based upon resource requirements and machine computational performance.

3.1.1 Users

Yes. Machines can be configured to not accept jobs when the machine is not idle or it is receiving keyboard input.

3.1.2 Disk Space

No. Support is planned for this in a future release.

3.1.3 System Load

Yes. Load Balancer takes into account the load and capacity of each machine before scheduling a job. It will also scale up the recorded load in anticipation of a job that it has just dispatched to a machine. This results in the machine not being swamped with jobs just as it becomes available.

3.1.4 Swap Space

Yes. Load Balancer dynamically checks the swap space on the machine.

3.2 Does the system provide checkpointing facilities for jobs?

No.

3.3 Is process migration between machines in the cluster supported?

No.

3.4 What is the impact on the machine owner?

This can be minimized. If an interactive user logs in, it is possible to configure Load Balancer to suspend, re-nice, or kill all jobs currently running on the machine. New jobs can also be prevented from running on the machine.

3.5 Are changes made in scheduling to account for fault tolerance?

Yes. Load Balancer will not submit a job to a machine that is not responding.

3.6 Does the system have the ability to suspend and/or resume jobs when resources change or become unavailable?

Yes. Load Balancer makes scheduling decisions dynamically based on whether the machine meets the desired requirements.

4.0 Configurability

4.1 Is it possible to place restrictions on the users of the system?
Yes. This can be done by assigning users priority levels, restricting the jobs they are allowed to run, the host they can run on, and the maximum number of jobs.

4.2 Does the system enforce job run time limits?
Yes. Job run time limits can be set for a group of hosts. A host can then be assigned to the group. This will effectively place a job run time limit on the jobs that execute on that host. Job run time limits can also be set for a group of applications.

4.3 Are priorities implemented for scheduling and resource allocation?
Yes. This is done by assigning priority levels. Users and applications can be assigned maximum priority levels which can restrict or enhance their execution time.

4.4 Can an exclusive CPU be made available if a job requires it?
Yes. Load Balancer can be configured so that if an application is running on a host, nothing else may run there until it finishes. You can also configure Load Balancer to run a certain application on a particular host. An application may be set so that only a specified maximum number of copies may run.

4.5 Can the user or system manager defer execution of a job?
Yes. Users may defer the execution time for up to a week. Fuller calendar-based job scheduling is planned in future releases.

4.6 Does the system have a Graphical User Interface (GUI)?
Yes, but it is a Windows front end to give PC users access to a UNIX network using a point-and-click interface. An X Window System interface is planned for a future release.

4.7 Is the system easy to learn?
Yes. No recompilation or relinking is necessary.

4.8 Can the user specify computing resources?
Yes. This is done interactively on the command line or the user may choose to accept the defaults. The user may specify a specific machine, a specific architecture, the amount of memory, the amount of swap space, and a specific operating system. Machines may also be assigned "features". Features include such things as tape drives, printers, licenses, etc. Users may specify which features their jobs require.

4.9 Can the user query the status of their job?
Yes. This is done interactively by the user using a utility supplied with Load Balancer.

4.10 Does the system return actual resource usage upon job completion?
Yes. This information will be send via e-mail and also stored in a central log file.

5.0 Dynamics of the System

5.1 Can the policies and queues of the system be reconfigured on the fly?

Yes. The configuration file must be edited to make changes, but the file is checked every 5 seconds for alterations.

5.2 Can the queues or machines be time of day sensitive?

Yes. Computers can be scheduled to come on-line at certain times of the day. Applications, users, and groups of machines can also be configured in this manner.

5.3 Can the machine be donated and withdrawn at will by the principal owner without the help of the system manager?

No. This feature will be updated in a future release.

5.4 Does the system possess any single point failure problems?

Yes. If the master daemons crash, they will restart when the machine comes back up. The jobs that were running will be lost unless they specify that they need to be rerun in the event of a crash.

5.5 Is the system fault tolerant?

No. If the master daemon crashes, the jobs that have not run will be lost. Any currently running job will run to completion, if its host machine is still running. The user must specify that their job be rerun if it crashes. This can be set as the default in the configuration file, if desired. Master daemon fault tolerance is being designed for a future release.

5.6 How does the system address security?

Load Balancer implements several security features. Programs cannot be run by the root. This attempts to stop major intrusions into the cluster by someone who has gained root access on one machine. The daemons all communication on secure port numbers. These ports are restricted to programs running as root. The clients that can connect to the cluster can be restricted to stop unauthorized machines from connecting.

Additional Limitations

- The fault tolerance of the master scheduling does not handle failure well. This flaw is to be corrected in a future release.

Additional Benefits

- Good attention paid to security.
- Very good documentation and manuals.

Sites running this software:

Amdahl
American Mathematical Society
ARCO Oil
Cirrus Logic
Defense Research Establishment
Fermilab

Motorola
Philips
Stratus Computer
Sun Microsystems

Software from:
Freedman Sharp and Associates, Inc.
508 - 1011 - 1st Street SW,
Calgary, Alberta,
Canada T2R 1J2
Phone (403)264-4822
FAX (403)264-0873
E-mail: lb@fsa.ca
See References: 9

LoadLeveler

Concept behind Implementation

LoadLeveler is a distributed, network-based, job scheduling program. LoadLeveler is a modified version of Condor from IBM, and briefly was referred to as "UniJES." It will locate, allocate and deliver resources from across the network while attempting to maintain a balanced load, fair scheduling, and an optimal usage of resources. The goal of this product is to make better use of existing resources by using idle equipment.

1.0 Is support provided for a heterogeneous computing environment?

Yes

1.1 Which platforms are supported?

IBM AIX, SGI IRIX, and SunOS.

1.2 How does the system know upon which architecture to execute the job?

The user must specify the architecture in the GUI, on the command line, or in a job submission file. If no specification is given, the architecture type default to that of the submitting machine.

1.3 What additional hardware and/or software is required for the system to work?

NFS or AFS.

2.0 Application Support

2.1 Is support provided for batch jobs?

Yes.

2.2 Is support provided for parallel jobs?

Yes, but only on static portions of the Scalable POWERparallel system.

2.3 Is support provided for interactive jobs?

No.

2.4 Is support provided for message passing between jobs?

No.

2.5 Can the system handle redirective jobs?

Yes. This is specified in the job submit file or on the command line when the job is submitted.

2.6 Can the files be accessed from the submitting machine?

Yes. This can be accomplished with either NFS or AFS.

3.0 Scheduling and Resource Allocation

3.1 What is the dispatching of jobs based upon?
All jobs are submitted to the most desirable host that meets all the requirements specified by the user.

3.1.1 Users
Yes. This is specified by the machine owner. The machine may always be available, never available, available during certain hours, or only available while the keyboard and mouse are idle.

3.1.2 Disk Space
The disk space is checked periodically. LoadLeveler will not dispatch a job to a machine that has less than the required amount of disk space.

3.1.3 System Load
Yes. LoadLeveler will attempt to dispatch a job to the machine with the lowest system load.

3.1.4 Swap
Yes. This is one of the requirement specified by the user.

3.2 Does the system provide checkpointing facilities for jobs?
Yes. Checkpointing is available to the user. They must specify in the job submission file that they want checkpointing. Then they must link their program with LoadLeveler's library. Once this is done, LoadLeveler will automatically handle all the checkpointing for the program.

3.3 Is process migration between machines in the cluster supported?
Yes.

3.4 What is the impact on the machine owner?
This changes depending upon how the machine owner configures the system. There are four possible settings: The machine is always available, the machine is never available, the machine is available between certain hours, and the machine is available whenever the mouse and keyboard are idle.

3.5 Are changes made in scheduling to account for fault tolerance?
Yes. The master scheduler will not submit jobs to a machine who's daemons are no longer active.

3.6 Does the system have the ability to suspend and/or resume jobs when resources change or become unavailable?
Yes.

4.0 Configurability

4.1 Is it possible to place restrictions on the users of the system?
Yes. The number of jobs per queue may be restricted. Users may also be assigned to particular

classes which have additional limits imposed upon them.

4.2 Does the system enforce job run time limits?
Yes. This may be specified by either the user or the system manager.

4.3 Are priorities implemented for scheduling and resource allocation?
Yes. Memory, features, machine, disk space, architecture, and operating system may be specified. The system manager may also set limits upon users.

4.4 Can an exclusive CPU be made available if a job requires it?
No. A machine can be configured so that only one job may run on it at a time.

4.5 Can the user or system manager defer execution of a job?
Yes. The user may opt to defer execution until a specific time and date.

4.6 Does the system have a Graphical User Interface (GUI)?
Yes. An AIXwindows based application is available to aid in the following commands:
- Building a job command file
- Editing a job command file
- Checkpointing a job
- Setting and changing the priority of a job
- submitting a job command file
- querying the status of a job
- Placing and releasing a hold on a job
- Cancelling a job
- View information regarding machines in the cluster
- Display the central manager
- Display public submit machines

4.7 Is the system easy to learn?
Yes. If the job does not request checkpointing, no relinking is required. The GUI will help with most tasks.

4.8 Can the user specify computing resources?
Yes. The user may specify these resources through the GUI, on the command line, or in a job submission file. The user may request memory, machine, architecture;, operating system and amount of disk space.

4.9 Can the user query the status of their job?
Yes. This is done by using utilities provided with LoadLeveler.

4.10 Does the system return actual resource usage upon completion?
Yes.

5.0 Dynamics of the System

5.1 Can the policies and queues of the system be reconfigured on the fly?
Yes. This can be done interactively through the AIXwindows GUI or by using a utility supplied with LoadLeveler that forces the master scheduler to reread all the configuration files.

5.2 Can the queues or machines be time of day sensitive?
Yes. Machine owners may specify the times that their machines are available for use.

5.3 Can the machine be donated and withdrawn at will by the principal owner without the help of the system manager?
Yes.

5.4 Does the system possess any single point failure problems?
No.

5.5 What conditions has the system made for fault tolerance?
Sites running this software: At the time of a crash, all jobs currently running will not be affected. If the master scheduler crashes, no new batch jobs will be scheduled for execution. The jobs currently executing will run to completion. If one of the executing machine's daemon's crash, the master scheduler will attempt to restart it. Since checkpointing is supported, the only loss of data that will result will be from the time of the last checkpoint.

5.6 How does the system address security?
The system relies on the underlying security of the UNIX operation system.

Additional Limitations
- Any job that uses the checkpointing facilities should avoid using the following system calls:
- Administrative(i.e. audit, swap, query)
- Signals
- Fork
- Dynamic loading
- Shared memory
- Semaphores
- Messages
- Internal timers
- Set user/group id

- All file operations must be idempotent - read only and write only file accesses work correctly, but programs which both read and write the same file may not.

- Only a limited number of platforms are supported.

Additional Benefits
- LoadLeveler builds adds features beyond the original Condor, including a GUI and NQS compatibility.

• LoadLeveler appears well suited for a production environment.

• The documentation accompanying LoadLeveler is very comprehensive with good illustrations.

Sites running this software:
Cornell Theory Center
Pennsylvania State University

Software from:
Fred Phillips
Vice President
IBM Business Partner Division
1212 Lake James Drive, Suite E2
Virginia Beach, VA 23464
Phone: (804)523-5664
Fax: (804)523-5647
E-mail: fredp@vnet.ibm.com
See References: 10,11

Load Sharing Facility (LSF)

Concept behind Implementation

LSF, formerly Utopia, provides two daemons which handle remote execution and job scheduling in a heterogeneous UNIX environment. Batch, interactive, and parallel execution functionality are built on top of these daemons. These packages are included with LSF. In a large cluster, some workstations may be overloaded while others sit idle. LSF was built with the principal of distributing the workload around a large cluster of desk-top computers. LSF operates by moving jobs around the cluster so that each machine has an even load.

1.0 Is support provided for a heterogeneous computing environment?
Yes.

1.1 Which platforms are supported?
ConvexOS, DEC OSF/1 - Ultrix, HP-UX, IBM AIX, SGI IRIX, and SunOS - Solaris.

1.2 How does the system know upon which architecture to execute the job?
The architecture is specified by the user using a command line argument. If the user neglects to select one, then execution defaults to the type of the submitting machine.

1.3 What additional hardware and/or software is required for the system to work?
Some type of shared file access is required (i.e. AFS, ATHENA, HFS, or HCS).

2.0 Application Support

2.1 Is support provided for batch jobs?
Yes. LSF provides a utility which supports batch jobs.

2.2 Is support provided for parallel jobs?
Yes. A generic interface is provided for parallel packages. PVM, TCGMSG, and parallel make are supported. LSF supports the same jobs control mechanisms for parallel jobs that it does for sequential jobs.

2.3 Is support provided for interactive jobs?
Yes. LSF provides a number of utilities which support interactive jobs, including a distributed make utility.

2.4 Is support provided for message passing between jobs?
Yes. Support is included for PVM, P4, Linda, TCGMSG, and DSM. Support is also include for communication and synchronization.

2.5 Can the system handle redirective jobs?
Yes.

2.6 Can the files be accessed from the submitting machine?
Yes. This done through a shared file system.

3.0 Scheduling and Resource Allocation
The scheduling mechanism for LSF does two things. The first is to find a suitable machine based upon user requirements. The second is to select the best machine out of the possible candidates. The batch facilities use a similar mechanism, but take into account the longer run times of batch jobs.

3.1 What is the dispatching of jobs based upon?
Jobs are dispatched to the host with the lightest load that satisfies the job resource requirements set by the user.

3.1.1 Users
Yes. LSF monitors the number of interactive sessions.

3.1.2 Disk Space
Yes. The available disk space in /tmp is monitored.

3.1.3 System Load
Yes. LSF monitors cpu run queue lengths, cpu utilization, memory, paging rate, I/O rate, and idle time.

3.1.4 Swap Space
Yes.

3.2 Does the system provide checkpointing facilities for jobs?
No.

3.3 Is process migration between machines in the cluster supported?
No.

3.4 What is the impact on the machine owner?
If the owner's machine is heavily loaded, the owner's processes will begin executing on another machine. Machines may also be configured so other processes will not begin execution if certain jobs or users are already running on the machine. LSF tries to minimize the impact while giving additional benefits to the machine owner.

3.5 Are changes made in scheduling to account for fault tolerance?
Yes. Only reachable hosts will receive jobs from the LSF scheduler.

3.6 Does the system have the ability to suspend and/or resume jobs when resources change or become unavailable?
Yes. LSF can suspend jobs when the situation on a host changes (i.e. certain users log in, certain jobs are running, load rises past a certain level, etc.) and then restart those jobs later.

4.0 Configurability

4.1 Is it possible to place restrictions on the users of the system?
Yes. The queues can be configured for different user groups with different policies and priorities.

4.2 Does the system enforce job run time limits?
Yes. Job run time limits may be imposed upon the queues using the batch system.

4.3 Are priorities implemented for scheduling and resource allocation?
Yes. Possible priorities include time, type of jobs, type of host, specific users and groups, and current resource loading conditions.

4.4 Can an exclusive CPU be made available if a job requires it?
Yes. Once an exclusive job is started on a host, not other batch or interactive job will be sent to that host until the exclusive job completes execution.

4.5 Can the user or system manager defer execution of a job?
Yes. Users may specify a starting time.

4.6 Does the system have a Graphical User Interface (GUI)?
No. A GUI will be added in a future release.

4.7 Is the system easy to learn?
Yes. There is no recompilation or relinking necessary. The user will be required to learn how to interact with the system, but it is fairly straightforward.

4.8 Can the user specify computing resources?
Yes. This is done as an argument on the command line. The specific machine, specific architecture, amount of memory, amount of swap space, specific type of operating system, and the amount of free disk space may be requested.

4.9 Can the user query the status of their job?
Yes. This is done interactively by using a utility supplied with LSF.

4.10 Does the system return actual resource usage upon job completion?
Yes.

5.0 Dynamics of the System

5.1 Can the policies and queues of the system be reconfigured on the fly?
Yes.

5.2 Can the queues or machines be time of day sensitive?
Yes. Queues and hosts can be activated or deactivated at different configurable time windows.

5.3 Can the machine be donated and withdrawn at will by the principal owner without the help of the system manager?
No. Threshold limits can be set to minimize the load on a machine. Machines can also be set so that they are submit only machines.

5.4 Does the system possess any single point failure problems?
No. If a host goes down, the only job loss that occurs are those jobs that were running on the crashed host. If all the machines in the cluster go down (i.e. power failure), all running jobs are lost, but unscheduled jobs will be dispatched as soon as the cluster is brought back on-line. In either case, all running jobs will be restarted on other hosts automatically.

5.5 Is the system fault tolerant?
LSF can recover if the master job handler is lost. All of the slave job handlers have an election to determine the new master. The election takes place until one finally asserts itself as the new master. So long as one slave job handler is still around, jobs will continue to be processed.

5.6 How does the system address security?
LSF offers user validation services beyond traditional UNIX security.

Additional Limitations

Additional Benefits
- The system is set up to handle large numbers of workstations.
- All tasks can be executed on the cluster, not just batch jobs.

Sites running this software:
Bell Northern Research
Pratt & Whitney
Western Digital
University of Washington
MITRE Corporation

Software from:
Platform Computing Corporation
203 College Street, Suite 303
Toronto, Ontario M5T 1P9, Canada
Tel: (416) 978-0458
FAX: (416) 978-0878
E-mail: info@platform.com
See Reference: 13

NQS/Exec

Concept behind Implementation

The Network Queueing System (NQS) was designed for NASA by Sterling software. Its purpose was to provide a good Unix batch and device queueing facility capable of supporting and tying together a diverse assortment of Unix based machines. NQS provides three types of queues. The first is the batch queue, which provides a mechanism to run programs. The second is the device queue, which provides for batch access to physical devices(i.e. printers, tape readers, etc.). The last type of queue is a pipe queue. A pipe queue exists to transport requests to other batch, device, or pipe queues at possibly remote machine destinations. NQS is geared more towards a production environment with supercomputers present. NQS/Exec adds an intelligent network batch job scheduling system that provides workload balancing across a heterogeneous Unix environment, and represents a natural progression from traditional NQS to clustered computing. This summary is based upon NQS/Exec.

1.0 Is support provided for a heterogeneous computing environment?

Yes.

1.1 Which platforms are supported?

HP-UX, IBM AIX,SGI IRIX, and SunOS - Solaris.

1.2 How does the system know upon which architecture to execute the job?

Specified by the user in a job submission file.

1.3 What additional hardware and/or software is required for the system to work?

NFS and NIS is recommended.

2.0 Application Support

2.1 Is support provided for batch jobs?

Yes.

2.2 Is support provided for parallel jobs?

No.

2.3 Is support provided for interactive jobs?

Yes.

2.4 Is support provided for message passing between jobs?

No.

2.5 Can the system handle redirective jobs?

Yes. This is specified when the job is submitted.

2.6 Can the files be accessed from the submitting machine?
Yes. This can be done through NFS.

3.0 Scheduling and Resource Allocation
"The scheduling algorithm is based on the memory size vs. usage, CPU usage percentage, and the run limit specified for the destined NQS batches. A job may be sent to a machine that has more memory and low CPU usage, but the batch queue run limit has been reached, therefore, NQS/Exec will try the next available machine in the list of destinations." - Tommie Bailey, Sterling Software

3.1 What is the dispatching of jobs based upon?
See section 3.0.

3.1.1 Users
No.

3.1.2 Disk Space
No.

3.1.3 System Load
Yes.

3.1.4 Swap Space
No.

3.2 Does the system provide checkpointing facilities for jobs?
No.

3.3 Is process migration between machines in the cluster supported?
No.

3.4 What is the impact on the machine owner?
Owners have no control of what runs on their machines. They may take a large performance hit depending upon what gets scheduled to execute on their machine.

3.5 Are changes made in scheduling to account for fault tolerance?
Yes. If NQS/Exec is unable to contact a machine, if will no longer schedule jobs on that machine.

3.6 Does the system have the ability to suspend and/or resume jobs when resources change or become unavailable?
No.

4.0 Configurability

4.1 Is it possible to place restrictions on the users of the system?
Yes. The number of jobs per user, the number of jobs per queue, and the job run time limits may

be limited depending upon the user. Users may also be excluded from groups that may access certain queues.

4.2 Does the system enforce job run time limits?
Yes. Job run time limits may be placed upon queues.

4.3 Are priorities implemented for scheduling and resource allocation?
No.

4.4 Can an exclusive CPU be made available if a job requires it?
No; however, machines may be configured in such a way that results in only one job executes at a time.

4.5 Can the user or system manager defer execution of a job?
Yes. A job's execution time can be delayed by using a command line option when the job is submitted. Both the date and the time of execution can be specified.

4.6 Does the system have a Graphical User Interface (GUI)?
Yes. NQS/Exec provides a Motif based GUI.

4.7 Is the system easy to learn?
Yes.

4.8 Can the user specify computing resources?
Yes. The amount of memory and the amount of disk space needed may be specified.

4.9 Can the user query the status of their job?
Yes. This is done by using a utility supplied with NQS/Exec.

4.10 Does the system return actual resource usage upon job completion?
Yes.

5.0 Dynamics of the system

5.1 Can the policies and queues of the system be reconfigured on the fly?
Yes.

5.2 Can the queues or machines be time of day sensitive?
Yes.

5.3 Can the machine be donated and withdrawn at will by the principal owner without the help of the system manager?
No.

5.4 Does the system possess any single point failure problems?
No.

5.5 What conditions has the system made for fault tolerance?
NQS/Exec will attempt to restart the daemon and the running jobs on any machine that crashes.

5.6 How does the system address security?
NQS/Exec provides files that detail who can run jobs on particular hosts and queues.

Additional Limitations

Additional Benefits

- Compatible with NQS.

Sites running this software:
Los Alamos National Labs
NASA Langley Research Center
University of Utah
Duke Power
BP Exploration
ORYX Energy Company
Phillips Petroleum
National Center for Supercomputing Applications

Software From:
Sterling Software
11050 White Rock Road #100
Rancho Cordova, CA 95670-6095
Phone - (916)635-5535 (800)862-3499
Fax - (916)635-5604
See Reference: 12

Conclusion

To determine which cluster management software is best suited for a given site, the needs and expectations of the users must be measured against what the different systems provide. All of the systems presented are the products of different usage philosophies.

Only the DQS and Condor systems have been tested at LaRC, and only DQS has been verified using PVM applications. However, the evaluations of all packages are as complete as documentation and vendor response have allowed.

Clustered and distributed computing are engaging new technologies that offer substantial computational power that can generally be culled from existing computers. Because this technology is still in development, the arrival of truly sophisticated cluster management systems that will hide most of the heterogeneous issues from the users is still in the future.

It is hoped that this summary of cluster management systems will enable the systems administrators to choose the system best suited for their users needs. Due to the rapid development pace of these systems, some of the features or limitations described within may be already addressed. Verification of functionality is recommended before a final decision is made. Efforts will be made to keep an updated on-line version of this report. Contact the authors for availability.

Acknowledgments

The authors wish to express their appreciation to John Liu of the Super Conducting Super Collider Laboratory and Doreen Cheng of NASA Ames Research Center for their thoughtful discussions about desirable cluster characteristics and evaluation criteria.

References

1. Litzkow, M., Livny, M., "Experience With The Condor Distributed Batch System," *Proceedings of the IEEE Workshop on Experimental Distributed Systems*, Huntsville, AL, October 1990.
2. Bricker, A., Litzkow, M., Livny, M., "Condor Technical Summary," University of Wisconsin - Madison, October 1991.
3. Litzkow, M., Solomon, M., "Supporting Checkpoint and Process Migration Outside The Unix Kernel.", University of Wisconsin - Madison, Usenix Winter Conference, San Francisco, California, 1992.
4. Litzkow, M., "Condor Installation Guide," University of Wisconsin - Madison, October 1991.
5. Livny, M., Pruyne, J., "Scheduling PVM on Workstation Clusters using Condor," University of Wisconsin - Madison, May 1993.
6. "The Distributed Job Manager Administration Guide," Army High Performance Computing Research Center, May 1993.
7. Revor, L. S., "DQS Users Guide," Argonne National Laboratory, September 1992.
8. Green, T. P., Snyder, J., "DQS, A Distributed Queueing System," Florida State University, March 1993.
9. Freedman Sharp and Associates Inc., "Load Balancer v3.3 Automatic Job Queueing and Load Distribution Over Heterogeneous UNIX networks," 1993.
10. International Business Machines Corporation, "IBM LoadLeveler: User's Guide.", Kingston, NY, March 1993.
11. International Business Machines Corporation, "IBM LoadLeveler: Administration and Installation.", Kingston, NY, March 1993.
12. Kingsbury, B. A., "The Network Queueing System," Palo Alto, CA., March 1993.
13. Zhou, S., "UTOPIA: A Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems," Computer Systems Research Institute, University of Toronto, April 1992.
14. Liebowitz, B. H., Carson, J. H., "Multiple Processor Systems For Real-Time Applications," The Sombers Group,

Prentice-Hall, Englewood Cliffs, New Jersey, 1985.
15. Wilmoth, R. G., Carlson, A. B., Bird, G. A., "DSMC Analysis in a Heterogeneous Parallel Computing Environment," AIAA Paper 92-2861, July 1992.
16. Cheng, D. Y., "A Survey of Parallel Programming Languages and Tools," Technical Report RND-93-005, NASA Ames Research Center, March 1993.
17. Birman, K. P., "The Process Group Approach to Reliable Distributed Computing," Cornell University, TR 91-1216, July 1991.
18. Ferstl, F., "CODINE Technical Overview," Genias, April, 1993.
19. Geist, A., Beguelin A., Dongarra J., Jiang W., Manchek B., Sunderam V., "PVM 3.0 User's Guide and Reference Manual," Oak Ridge National Laboratory, ORNL/TM-12187, February, 1993.
20. Grant, B. K., Skjellum A., The PVM Systems: An In-Depth Analysis and Documenting Study - Concise Edition, Lawrence Livermore National Laboratory, September 1992.
21. Benway, R., "A User's Experience with Increasingly Large PVM Klusters", Presented at the *PVM Users' Group Meeting*, Knoxville, Tennessee, May 1993.
22. "C-Linda Reference Manual", Scientific Computing Associates, Inc.,1992.
23. Butler, R., Lusk, E., "User's Guide to the P4 Programming System," Argonne National Laboratory Technical Report ANL-92/17, 1992.
24. Kolawa, A., "The Express Programming Environment," Presented at the *Workshop on Heterogeneous Network-based Concurrent Computing* Tallahassee, FL, October 1991.
25. Rinaldo, F. J., Fausey, M. R., "Event Reconstruction in High-Energy Physics," *Computer*, June 1993, pp. 68-77.
26 Allen, J. et al, "Physics Detector Simulation Facility System Software Description," SSCL, SSCL-SR-1182, December 1991.
27. Turcotte, L.H., "A Survey of Software Environments for Exploiting Networked Computing Resources," MSU-EIRS-ERC-93-2, Mississippi State University, June 1993.
28. GENIAS Software, "CODNE Reference Manual," Genias, May, 1993.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	October 1993	Technical Memorandum

4. TITLE AND SUBTITLE
A Comparison of Queueing, Cluster and Distributed Computing Systems

5. FUNDING NUMBERS
WU 505-90-53-02

6. AUTHOR(S)
Joseph A. Kaplan
Michael L. Nelson

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23681-0001

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
National Aeronautics and Space Administration
Washington, DC 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
NASA TM-109025

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified–Unlimited

Subject Category 61,62

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Using workstations clusters for distributed computing has become popular with the proliferation of inexpensive, powerful workstations. Workstation clusters offer both a cost effective alternative to batch processing and an easy entry into parallel computing. However, a number of workstations on a network does not constitute a cluster. Cluster management software is necessary to harness the collective computing power. In this paper, we compare a variety of cluster management and queueing systems: Distributed Queueing Systems (DQS), Condor, LoadLeveler, Load Balancer, Load Sharing Facility (LSF - formerly Utopia), Distributed Job Manager (DJM), Computing in Distributed Networked Environments (CODINE) and NQS/Exec. The systems differ in their design philosophy and implementation. Based on published reports on the different systems and conversations with the system's developers and vendors, a comparison of the systems are made on the integral issues of clustered computing.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Distributed Computing; Clusters; Workstations; Queueing Systems	48
	16. PRICE CODE A03

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	